Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/MR/7441--96-7719

Survey of Spatial Topology: Issues and Approaches

David K. Arctur
John F. Alexander

University of Florida
Gainesville, FL

Miyi J. Chung
Maria A. Cobb
Kevin B. Shaw

Mapping, Charting, and Geodesy Branch
Marine Geosciences Division

September 20, 1996

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
September 20, 1996

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Survey of Spatial Topology: Issues and Approaches

5. FUNDING NUMBERS
Job Order No. 5745137A6
Program Element No. 0603207N
Project No.
Task No. R-1987
Accession No. DN 153-135

6. AUTHOR(S)
David K. Arctur*, John F. Alexander*, Miyi J. Chung,
Maria A. Cobb, and Kevin B. Shaw

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER
NRL/MR/7441--96-7719

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Chief of Naval Research
Code 824
800 North Quincy Street
Arlington, VA 22217-5050

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
*University of Florida, Gainesville, FL

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report documents our current experience and understanding with respect to the design of the Object-Oriented Vector Product Format prototype viewer/editor application for managing graphical primitive objects and operations while maintaining full spatial topology. This is an interim progress report within the Object-Oriented Database Exploitation Within the Global Geospatial Information and Services (GGIS) Data Warehouse project, sponsored by the Defense Mapping Agency (DMA). The goal of the overall project is to investigate, through research and prototyping efforts, the potential impact of object-oriented technology on DMA's GGIS modernization program.

14. SUBJECT TERMS
Tactical oceanography, dynamical oceanography, physical oceanography, electronic/electrical engineering

15. NUMBER OF PAGES
14

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Introduction

This report documents our current experience and understanding with respect to the design of the Object-Oriented Vector Product Format (OVPF) prototype viewer/editor application for managing graphical primitive objects and operations while maintaining full spatial topology. This is an interim progress report within the Object-Oriented Database Exploitation Within the Global Geospatial Information and Services (GGIS) Data Warehouse project, sponsored by the Defense Mapping Agency (DMA). The goal of the overall project is to investigate, through research and prototyping efforts, the potential impact of object-oriented (OO) technology on DMA's GGIS modernization program (other reports include Shaw 1995, Arctur1995a, Arctur 1995b, Chung 1995).

This report provides a basic description of topology through a review of current literature, with references to node graph theory. An overview of design issues and implementation of full spatial topology in OVPF is then presented. Examples of current usage are drawn from a commercial Geospatial Information and Services (GIS) software product, Arc/Info by Environmental Systems Research Institute (ESRI 1994), both to help clarify some of the concepts and to illustrate features that a complete GIS product must include in its design. Arc/Info is chosen because it is a mature software product for manipulating georelational data, rather than simply a specification for the structure of such data. This will provide a useful context for comparison and discussion of VPF specification (DMA 1993), and the OVPF application.

Key Concepts of Graph Theory and Topology

Geographic features have both metric properties (location, length, and other attributes, including nonspatial attributes), and topological properties (adjacency and connectivity), as described in Laurini (1994). The metric information is used for:

- drawing maps of nodes, lines, and surfaces;
- differentiating between duplicate edges;
- locating intermediate points on edges;
- measuring distances, perimeters, shapes, areas, or volumes; and
- determining positions of entities.

Various degrees of topological information may be needed, depending on the application and resources available. The topological information is used for:

- tracing and buffering analysis;
- determining the character of adjacent area units;
- automating some error detection procedures;
- making data updates more feasible by separating the metric information from the structural;
- facilitating aggregation of primitive spatial units into larger units;
- providing a basis for automation in map matching and transformations; and
- facilitating spatial reasoning.

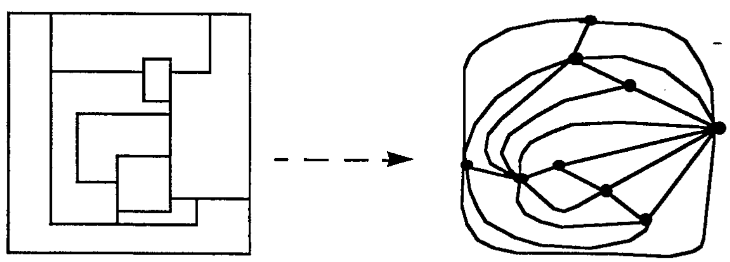
Basic understanding of graph theory is a prerequisite to a description of topology. For purposes of this report, a graph is a combination or network of line segments (Laurini 1994). In a stricter mathematical setting (Harary 1969), a more general definition is usually assumed, but Harary acknowledges that graph and other related terms are frequently defined according to the frame of reference in which they are used. As we are concerned with applications to digital cartography, we will start with Laurini's definition here.

Figure 1 below shows two alternate forms of graphs for maps. In Figure 1(a), spatial areas from the diagram on the left become nodes in the graph on the right, while in Figure 1(b), the intersections of spaces on the left become nodes in the graph on the right. Both approaches have useful applications, to be developed shortly. In either case, absolute positioning and other metrics from the maps are not retained in the graphs; only the structural relationships of connectivity and contiguity.

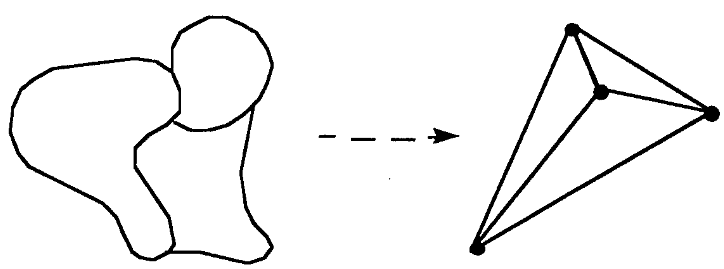
With respect to the graphs in Figure 1, we can define the following terms (most commonly used terms are emphasized):

1. The intersections or end points of lines are usually referred to as vertices, nodes, or 0-cells (as in zero-dimensional).
2. The lines are generally called edges, arcs, links, chains, or 1-cells.
3. The vacant spaces between or outside arcs are also called faces, polygons, regions, or 2-cells. Arc/Info actually uses the term region in a stricter sense to represent user-specified groups of polygons.

In Figure 1 all connections between the graph lines are planar, that is, all crossings occur in the same plane and result in a node. Graphs in general need not be planar, as in a road network with underpasses and overpasses, but this will be seen to violate the conditions of topological consistency.



(a) Map is transformed into graph in which graph nodes represent geometric spaces (Harary 1969, p. 6)



(b) Map is transformed into tesselation graph in which each polygon has minimum of three sides and three vertices (Laurini 1994,p.187)

Figure 1. Alternate graphs of geometric spaces

A line segment with at least one node that is not connected to another line segment is called a subgraph (see Figure 2). Arc/Info refers to the unconnected end of a subgraph as a dangling node. While dangling nodes can be an indication of data errors in digitizing or graph-building from the map, they can also be perfectly valid, as in the case of a cul-de-sac in a street network. An isolated, unconnected node in space is not considered to be part of any graph.

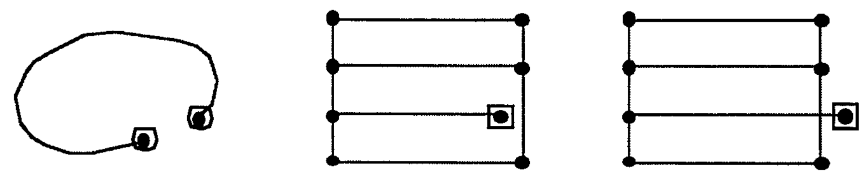


Figure 2. Examples of subgraphs and dangling nodes (ESRI 1994)

There are five basic rules of topology (Corbett 1979). We are substituting VPF’s node-edge-face terminology for that used by Corbett. These rules are illustrated in Figure 3 below:

1. Every edge is bounded by two nodes.
2. Every edge is bounded by two faces.
3. Every face is bounded by edges and nodes.
4. Every node is bounded by edges and faces.
5. There are no intersections that are not nodes.

Under these rules, isolated points will still be points, but points that are topological junctions will be called nodes. Similarly, disconnected or unrelated linear features are not edges ; this term refers only to bounding edges of faces. Within the VPF specification, an isolated point may also be an entity node, and have a topological relationship with its containing face. What Laurini and Arc/Info refer to as a node is what VPF defines as a connected node, for distinction from entity nodes.

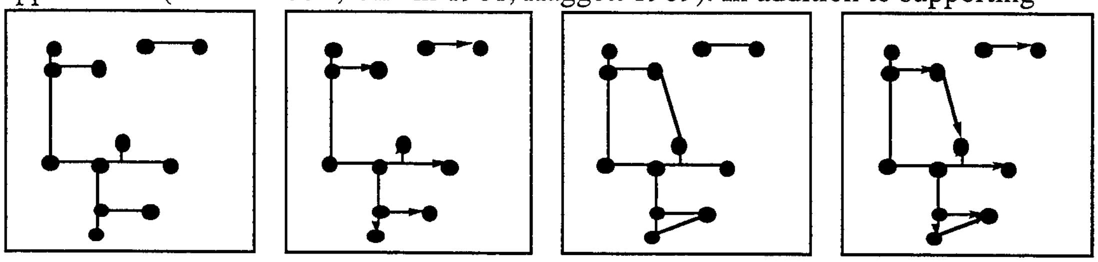
Figure 3. Topological relationships (Laurini 1994, p. 187)

There are some additional classifications of graphs that can be made, such as (Laurini 1994, p.178):

- If a graph has no loops, circuits, or cycles, it is called a tree graph, or just a tree (Figure 4a).
- If a graph has directed arcs but no circuits, it is called a directed acyclic graph (Figure 4b).
- Graphs with circuits, that is cyclic graphs, have at least one vertex connected to itself without the need for traversing one edge in both directions (Figure 4c).
- Both cyclic and acyclic graphs may be directed or oriented, as indicated by arrows.

Topology can be seen to represent a specific set of constraints applied to graphs to facilitate the use of graph theory and algorithms in addressing questions related to

connectivity, contiguity, and proximity among areal, geometric features. A number of interesting and sophisticated techniques have been developed for traversing and manipulating graphs (Lau 1989, Harary 1969), which serve well in many cartographic applications (Laurini 1994, Unwin 1981, Haggett 1969). In addition to supporting



Unoriented (e.g., railways) (a) — Oriented (e.g., rivers) (b) — Unoriented with circuits (e.g., highways) (c) — Oriented with circuits (e.g, air routes) (d)

Figure 4. Some types of graph patterns (Laurini 1994, p. 179)

connectivity-oriented queries and processing, the topology model also affords certain kinds of data integrity checks (Laurini 1994), which will be further discussed below.

Another key term to be introduced in relation to topology is coverage. This generally refers to the set of all geospatial features for which topological relationships have been determined. Arc/Info defines a coverage as:

> "... the framework for vector data storage in Arc/Info. It generally represents a single set of geographic objects such as roads, parcels, soil units, or forest stands in a given area. A coverage supports the georelational model—it contains both the spatial (location) and attribute (descriptive) data for geographic features." (ESRI 1994)

VPF uses *coverage* in a slightly broader sense to refer to a grouping of several related feature classes for which topological relationships have been established within a specified geographic area.

Not all cartographic situations can be easily handled topologically, as noted in the case of depicting underpasses or overpasses in transportation networks. Different levels of topology have been considered in various GIS databases and applications. VPF categorizes these as:

- *Level 0*- (boundary representation only; no intersections of lines or areas considered)
- *Level 1*- (nonplanar graph, commonly referred to as "spaghetti")
- *Level 2*- (planar graph, commonly referred to as "network topology")
- *Level 3*- (full spatial topology, with no overlapping areas or line segments)

Even within VPF specifications, different levels of topology may be used for a given database, library or coverage. For example, Digital Nautical Chart uses Level 3 topology for all coverages except LIBREF and TILEREF libraries, which use Level 2.

Further discussion on other aspects and applications of the topology model will be found in the sections following, beginning with a survey of key historical developments in this field.

Review of Alternative Proposals for Dealing with Topology

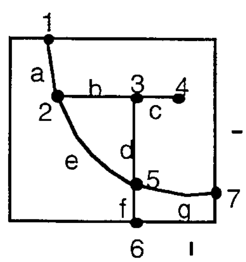
The earliest data organization approaches for representing digital cartographic data did not encode topology, but simply stored all individual spatial units as separate unconnected elements, as in the polygon model of the SYMAP software, developed primarily at Harvard University in the late 1960s. In these nontopological data models, only positional

information is stored, such as the point coordinates and the list of points forming each polygon.

The first topologically explicit data model was the line segment structure, developed and used by the U.S. Census Bureau for the 1970 census. This DIME (Dual Independent Map Encoding) format had a data record for each line segment, including from-node, to-node, left-polygon, and right-polygon, as well as the coordinates of the nodes and names of the polygons. There were no intermediate "shape" points stored, so that a single record could handle both geometry and topology. Only line segments were stored, so polygons had to be laboriously assembled from the data encoded in the line segment records.

The *chain model*, sometimes known as the POLYVRT structure, extends the line segment structure by encoding intermediate points along lines, as well as the line end points. One table contains the list of points or nodes (and their coordinates) making up each chain, while another table lists the end-nodes and adjacent faces for each chain. This system was used at Harvard University about 10 years ago. The U.S. Geological Survey's (USGS) first digital line graph (DLG) concept was encoded in a similar way, with additional information such as centroid coordinates for areas.

Connectivity Map



Link	From-node	To-node
a	1	2
b	2	3
.	.	.

(a) Link list

Node	1	2	3	4
1	-	1	0	0
2	1	-	1	0
3	0	1	-	1
.	.	.	.	-

Node	Connected to
1	2
2	1,3,5
3	2,4,5
.	. . .

(b) Connectivity matrix and node list

Figure 5. Recording data for *connectivity*.
Contiguity follows the same principle with areas identified instead of nodes.
(Laurini 1994, p.209)

Connectivity and *contiguity tables* provide a straightforward way of representing topology (Figure 5 above). Connectivity can be handled by link-node lists, showing *from-node* and *to-node* labels, with a separate table for the lines' coordinates (Figure 5a). An alternative to these lists is the use of a connectivity matrix and separate node coordinate list (Figure 5b). The same idea is applied to a contiguity table, in which the *left-* and *right-polygons* for each link would be identified. Note that these can be used for either directed or undirected graphs. Lists seem to be better for some purposes than matrices, and are certainly less sparse. However, some arithmetic operations are performed more easily on matrices than on lists.

The more recent TIGER (Topologically Integrated Geographic Encoding and Referencing) model ((Marx 1986); see Figure 6), developed jointly by the USGS and the U.S. Census Bureau, grew considerably from the agencies' experiences with DIME and DLG. TIGER provides a rich data model that supports full spatial topology based on the work of (Corbett 1979).

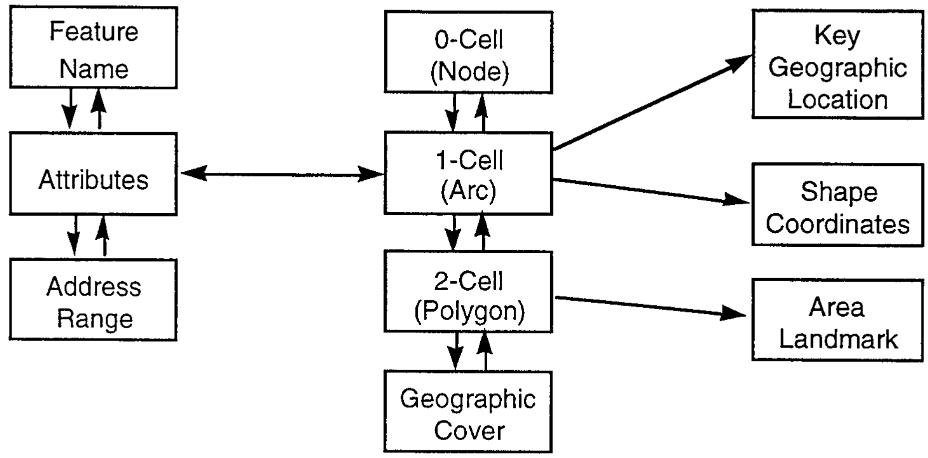


Figure 6. TIGER Structure
(Marx 1986)

Shortly after TIGER's release with the published 1990 census data, another data model standard was released by the USGS and U. S. Census Bureau, the Spatial Data Transfer Standard (SDTS), (National Institute of Standards and technology 1992). SDTS was initially developed to make the TIGER database easier to use, and following the public release of SDTS in 1991, the TIGER database has been provided in that format. The most notable feature about SDTS is its oo flavor. It cannot be said to be an oō system, as it does not incorporate behavior, but simply provides a data structure specification. However, it supports many of the characteristics of oo design, including a class hierarchy of spatial objects, generalization (abstract classes), aggregation (composite classes), and association (objects linked by phenomena rather than by definition). Because of its flexible approach, the U. S. Census Bureau was able to completely define the TIGER data entities in terms of SDTS spatial objects, while retaining certain key elements unique to the TIGER model. The SDTS model supports both topological and nontopological (geometrical) spatial entities, and in fact provides classifications for each of several degrees of completeness in topological information that may be available (see Figure 7 for a partial list). It appears the USGS and the U. S. Census Bureau have created considerable growing room for future digital products (Fegeas 1992, Lazar 1992, Davis 1992, Szemraj 1992).

	One-Dimensional Spatial Objects
Line Segment	An object representing a straight line connecting two points.
String	An ordered sequence of connected, nonbranching line segments (a string may intersect itself or other strings).
Arc	A curve that is defined by a mathematical function.
Link	A topological connection between two nodes.
Chain	A directed, nonbranching sequence of nonintersecting line segments and/or area bounded by nodes.
- Complete Chain	A chain that explicitly references left and right polygons, and beginning and ending nodes.
- Area Chain	A chain that references left and right polygons, but not beginning and ending nodes.
- Network Chain	A chain that references beginning and ending nodes, but not left- and right-polygons.
Ring	A sequence of nonintersecting chains, strings, and/or arcs that close to form the boundary of an area.
- G-Ring	A ring created from strings and/or arcs.
- GT-Ring	A ring created from complete and/or area chains.

Figure 7. Primitive line-vector object names and definitions in SDTS (Davis 1992, p. 324)

While many other frameworks may exist, these should suffice to illustrate the main issues involved and approaches currently being used. The next section focuses on current implementations of topology.

Current Implementations of Topological Structure

For simplicity of comparison, this and following sections present just the frameworks used in VPF and Arc/Info. Both use directed cyclic graphs to represent edges and faces, and assign a unique integer identifier to each. The internal organization of topology data used by Arc/Info to represent these structures is similar in some ways to VPF. For example, each arc (edge) knows its start- and end-nodes, as well as its left- and right-polygons (faces). However, Arc/Info stores the location coordinates of arcs in a separate file from the topological relationships, while VPF specifies that a single file contains both location and topology data for each type of topological primitive (nodes, edges, and faces). In VPF, an edge also knows its left edge from the start-node, and its right edge from the end-node. This is part of VPF's winged-edge topology, illustrated in Figure 8. And while Arc/Info stores an arc's length with its topological relations, VPF does not specify that an edge knows its length.

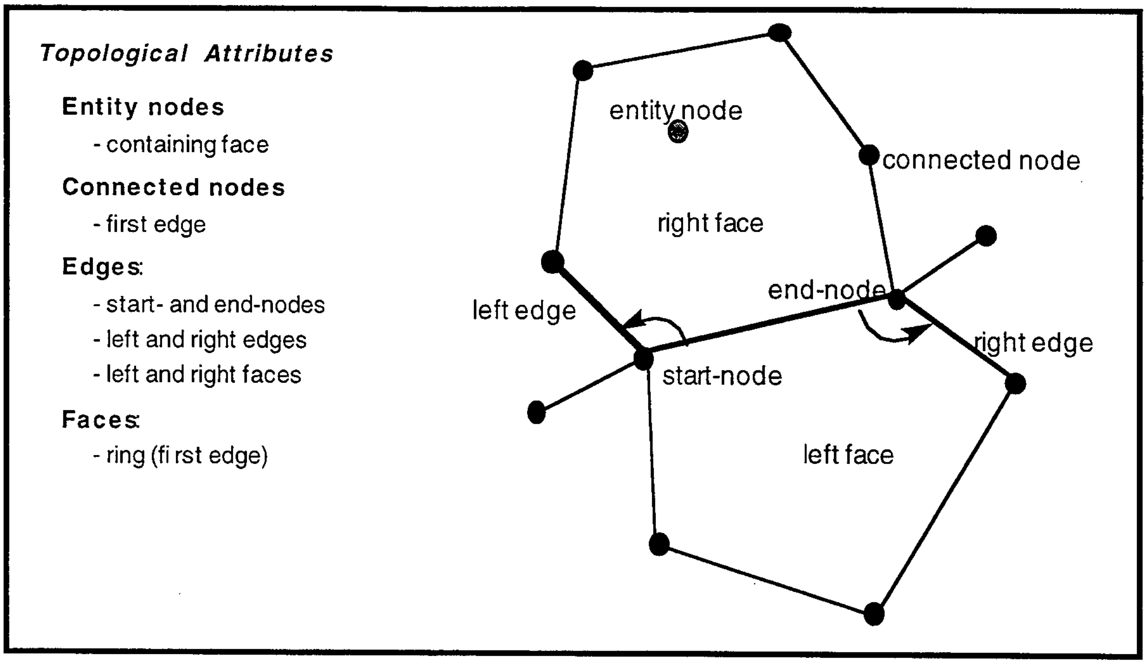


Figure 8. Winged-edge topology relationships in VPF (Cobb 1995)

Topological Operations

Most GIS software products today provide two principle operations related to construction of topological structures: *build* and *clean*. The job of *build* is to take the spaghetti of lines digitized from a map (Laurini 1994, p.190) and:

- Determine the containing face of entity nodes.
- Determine where lines cross to establish the connected nodes.
- Create all the separate edges.
- Assemble ordered sets of nodes to make nonoverlapping faces.
- Associate edges with their nodes.
- Run the *clean* operation to address the more obvious geometric and topological errors.

In carrying out these steps, the five principle rules of topology provide the main guidelines for the algorithms, but various errors can occur in either the digitizing or the topology-building processes. Some *geometric error conditions* include (Laurini 1994, p.191):

- A node is missing or misplaced.
- An edge is missing or misplaced.
- An edge has a bad shape or too many (or too few) points on it; or coordinates are missing or incorrect.
- A node has more than one position.

Some topological conditions include:

- Unconnected edges exist.
- A face has a gap between two edges that is it is not closed.
- Duplicate edges are present.

- A face has more than one or no reference point associated with it.
- A node has only one or two edges, rather than at least three.
- A face may be missing.

As Laurini points out (p.191),

> "At times, unless external data are available for checking, it may not be easy to validate whether a condition is inherently geometrical or topological, or indeed, whether or not it even represents an error."

In identifying and attempting to correct some of these errors, the clean operation in Arc/Info takes into account numerous user-specified tolerances, including (ESRI 1994):

- *Fuzzy tolerance* - an extremely small distance used to resolve inexact intersection locations due to limited arithmetic precision of computers.
- *Dangle length* - the minimum allowed length for a dangling arc.
- *Node snap tolerance* - the minimum distance within which two nodes will be joined (matched) to form one node.
- *Arc snap tolerance* - the distance within which a new arc will be extended to intersect an existing arc.
- *Weed tolerance* - the minimum allowable distance between any two vertices along an arc (used to reduce the number of coordinates in an arc).
- *Grain tolerance* - further controls the number of vertices in an arc and the distance between them.

The foregoing discussion provides an indication of the types of issues and difficulties involved in creating and maintaining topological structures based on geographic source data. The next section summarizes the main points of our design for an oo representation for topology.

Topology Design Issues in an Object-Oriented Environment

As we introduce OVPF's oo framework, it is important to point out some basic distinctions between the OO approach and the georelational approach. In the georelational approach, all data are organized into tables of rows and columns, and a given row's identity is based on its primary and foreign key values. Query and analysis procedures are external to the data tables, and maintained independently of them. As the data structures and the procedures become increasingly complex, keeping them in sync through changes becomes increasingly difficult.

An *object* is somewhat analogous to a row in a relational table. An object's attributes correspond to the columns of a relational table. But in the OO paradigm, we think of *objects* as data elements that know what kinds of operations they can perform and how to perform them. Basically, the query and analysis procedures are distributed among all the objects, rather than being external to, and manipulating, isolated data elements. This results in much tighter coupling between a given data structure and the procedures that are concerned with it, which facilitates easier and more rapid maintenance to keep them in sync through changes in the system. Another distinction is that each object has a unique identity, and holds direct pointers to its component objects' identities, resulting in inherent support for complex object webs without the need for the application software to maintain primary and foreign key values. We will now see how these kinds of differences affect our design of spatial topology in an OO framework.

From the literature, it appears that graph mathematics have a lot to offer, but that topology is computationally expensive to determine, debug, and maintain. Thus, topology

should be physically stored within the spatial data model if it is desired. The question of the possible implementations then depends on:

- How much topological information should be stored?
- When and how should it be determined?

It seems safe to say that *each feature object should be capable of holding onto direct pointers to its connected and adjacent neighbors*, depending on its status as a node, edge, or face object, and on the context of the application. It also seems safe to say that *these pointers should be computed as seldom as possible, and within as limited a geographic scope as possible*, and still support accurate, topologically-based queries and analysis.

The topology design is also complicated by the issue of *scale*, which is not addressed in any of the previously cited data models. For example, a feature that may be an *area* object at one scale could be a *point* object at another scale, or *not even shown* at still another, as we "zoom out" in relation to the map. Presently, VPF stores these different representations in separate *libraries* of coverages, but DMA has indicated dissatisfaction with this approach, as it imposes arbitrary restrictions on some cross-feature queries. Shall we consider all features to be instantiations of their largest-scale (closer-in) representation, and only change the display attribute as the scale changes? Under what conditions shall the topology data change as the map's scale and features' representations change? It could impose considerable, seemingly unnecessary overhead at smaller scales (viewing from a more distant vantage point) to keep track of all features, whether displayed or not. However, the fine detail may be required, regardless of scale, depending on the nature of a user's query (e.g., "zoom in on the bridge that is four miles up a small stream from buoy 'A'.")

Finally, the OVPF design for handling topology must take into account the need to export feature data back to relational VPF files. It is expected that OVPF will be used to import, edit, and export relational VPF data to support the current and future needs of DMA's customer base. Certainly from this requirement, OVPF must include at least the topological attributes already specified in VPF.

An Object-Oriented Topology Implementation

In OVPF, we have introduced an object known as a DrawOrder. This is a very simple structure whose inspiration is drawn from Digitalk's Smalltalk/V for OS/2 Presentation Manager (Digitalk 1989, p.464). The structure contains a variable-length array of bytes (the contents attribute). Each DrawOrder's contents array has the following organization:

- *opcode* -- a single byte whose integer value (0 - 255) represents an operation code, such as set polyline, continue line, set color, etc.
- *byte length* -- a single byte whose integer value (0 - 255) represents the number of bytes remaining in this draw order.
- *data bytes* -- the bytes whose integer or floating-point values represent the location points, the line-color index, etc., for this draw order.

The byte-array contents from several DrawOrders can be concatenated into a single DrawOrder to include an arbitrary number of instructions for displaying complex graphical objects. This structure is not only versatile, it is very compact and efficient for representing location coordinate data.

Even without the need to manage spatial topology, *DrawOrders* are useful objects for primitive graphical data and operations. Supporting VPF graphical primitives with full spatial topology represents a refinement of this definition, so these primitives are implemented by subclassing the DrawOrder class. A straightforward example would be to have *EntityNode, ConnectedNode, Edge, Face,* and *Ring* classes defined as direct subclasses of DrawOrder. In this way, each subclass would inherit the *contents* attribute, and add its own specific topological attributes as needed. It is also important, however, for each graphical object to hold onto a collection of the VPF *feature objects* that use that

graphical object. This is handled in OVPF by defining the *TopologicalStructure* class as a subclass of DrawOrder and as a superclass of each VPF graphical primitive class (see Figure 9). The *TopologicalStructure*'s *features* attribute is handled as *a collection of features* because a given unique graphical primitive may be used to represent any number of VPF features. Each feature object holds onto an identity-pointer to its corresponding collection of graphical primitive objects (see (Arctur 1995a)), thus enabling both features and primitives to have access to each other.

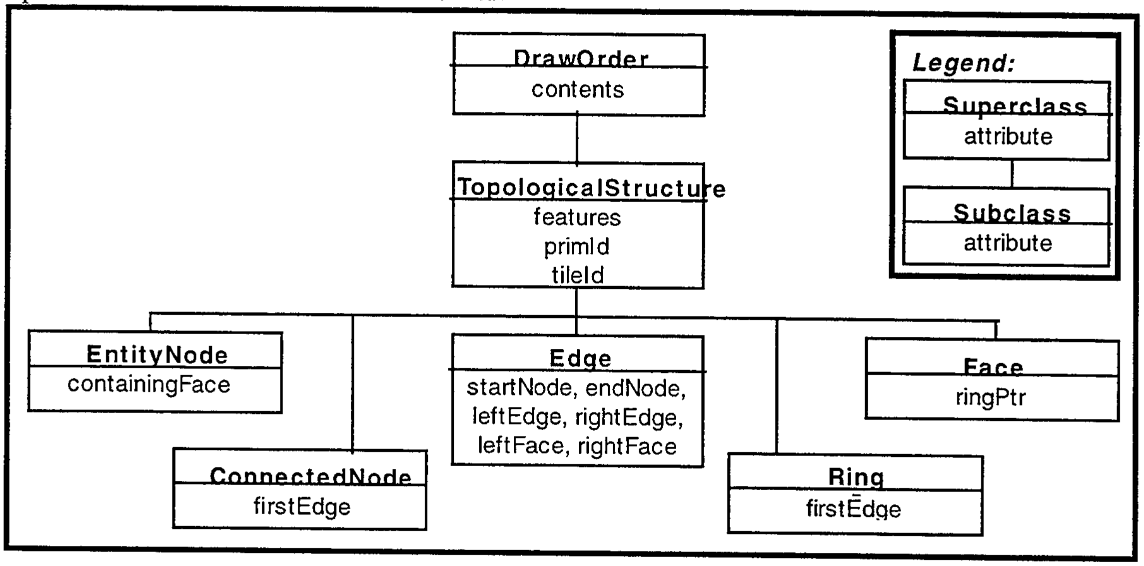


Figure 9. Object hierarchy for representing VPF graphical primitives with spatial topology

In addition, the primId (primitive ID) and tileId attributes of *TopologicalStructure* are inherited by each subclass, providing a holding place for primary-key data from the relational VPF files. For simplicity of supporting both import and export operations with the relational VPF data, the primId, tileid, and topological attributes are assigned the primId value of the corresponding graphical primitive objects, rather than unique object-identity pointers. As features are added, deleted, and moved with respect to each other, these primId values are maintained just as they would be in a relational GIS framework.

Presently, all source data comes to OVPF from relational VPF databases. At this stage in our prototype development, we assume that all feature attribute, location, and topological relationships in the source data are initially correct. Thus, we can focus our attention on developing full *build* and *clean* topological support in a step-wise manner, beginning with simply *maintaining* topology during isolated feature changes. We now have the capability to interactively add, delete, and change location coordinates of a single point, line, or area feature at a time within a given tile, while maintaining correct topological relationships with adjacent and contiguous features (Chung 1995). This is handled through a graphical user interface that requires the user to accept and commit changes to each topological relationship. The next step will be to support changes to features that span tile boundaries. After that, we will build support for *batch changes* to topology, such as from merging multiple coverages. At that stage, separate noninteractive *build* and *clean* procedures will be implemented. For efficiency and performance, we will continue to use localized, interactive techniques for maintaining topology whenever possible, and use the batch-oriented *build* and *clean* procedures only when necessary.

Summary

Topology is not always needed in GIS applications. Geometric information is not always important in GIS applications. Each of these has significantly different implications

for data model design, perhaps more so in an oo model, since the analytical behavior is more closely associated with the structural data objects. In the present OVPF prototype, we have designed and implemented a versatile and efficient framework for representing graphical primitives in general, as well as for handling full spatial topology. We have implemented the capability to add, delete, and change coordinate locations of point, line, and area features, while maintaining correct topological relationships. This framework can easily be extended and even reorganized as needed to support future enhancements, such as to provide batch-mode *build* and *clean* operations for merging coverages.

Acknowledgement

We wish to thank our sponsor, the Defense Mapping Agency Mr. Jim Krause and Mr. Jake Garrison as program managers, for sponsoring this research.

References

Arctur, D. K., K. B. Shaw, M. J. Chung, and M. A. Cobb (1995), "OVPF Report: Object-Oriented Database Design Issues," Interim Project Report to DMA, June 1995.

Arctur, D. K., K. B. Shaw, M. J. Chung, and M. A. Cobb (1995), "OVPF Report: Evaluation of Illustra Hybrid Object-Relational DBMS," Interim Project Report to DMA, July 1995.

Chung, M. J., M. A. Cobb, K. B. Shaw, and D. K Arctur (1995), "OVPF Report: Network Investigation Results," NRL Memorandum Report MR/7441--95-7713, July 1995.

Cobb, M. A., and K. Shaw (1995), "FY95: Object-Oriented VPF (OVPF) Project Status," notes from presentation by Naval Research Laboratory to Defense Mapping Agency, July 26, 1995.

Corbett, J. P. (1979), Topological Principles in Cartography, Technical Report No. 48, U.S. Bureau of the Census, Washington, D.C., U.S. Government Printing Office.

Davis, B. A., J. George, and R. Marx (1992) "TIGER/SDTS: Standardizing an Innovation," Cartography and Geographic Information Systems 19, 5: 321-327.

Digitalk, Inc. (1989), Smalltalk/VPM Tutorial and Programming Hankbook. Los Angeles:, CA, Digitalk, Inc.

Defense Mapping Agency (1993), Military Standard: Vector Product Format, Draft Document No. MIL-STD-2407, Fairfax, VA, DMA.

Environmental Systems Research Institute (1994), Arc/Info Users Guide, Release 7, Redlands, CA, Environmental Systems Research Institute.

Fegeas, R. G., J. Cascio, and R. Lazar (1992), "An Overview of FIPS 173, The Spatial Data Transfer Standard," Cartography and Geographic Information Systems 19, 5: 278-293.

Haggett, P; and R. Chorley (1969), Network Analysis in Geography, New York, NY, St. Martin's Press.

Harary, F. (1969), Graph Theory, Menlo Park, CA, Addison-Wesley.

Lau, H.T. (1989), Algorithms on Graphs, Blue Ridge Summit, PA, TAB Books.

Laurini, R. and D. Thompson (1994), Fundamentals of Spatial Information Systems, New York, NY, Academic Press.

Lazar, R. A. (1992), "The SDTS Topological Vector Standard," Cartography and Geographic Information Systems 19, 5: 296-299.

Marx, R. W. (1986), "The TIGER Aystem: Automating the Geographic Structure of the United States Census," Government Publications Review 13: 181-201.

National Institute of Standards and Technology (1992), Spatial Data Transfer Standard (SDTS), FIPS Pub. 173, Springfield, VA, National Technical Information Service.

Samet, H. (1990), The Design and Analysis of Spatial Data Structures, Reading, MA, Addison-Wesley.

Shaw, K. B. (1995), "Object-Oriented Database Exploitation Within the GGIS Data Warehouse: Initial Report to DMA,"NRL Report FR/7441--95-9639, Stennis Space Center, MS.

Szemraj, J. A., (1992), "TIGER/SDTS Topology," Cartography and Geographic Information Systems 19, 5: 328-331.

Unwin, D. (1981), Introductory Spatial Analysis, New York, NY, Methuen & Co.